3N-7

自律型 Web サービス　メッセージング基盤の開発

木村　泰輔†　　二宮　良太†　　平本　真道†　　大谷　真†

湘南工科大学†

1. はじめに

自律型 Web サービス(AWS;Autonomous Web Services)の基盤部は非同期メッセージング機能である。現在までに様々な Web サービスの技術の標準化が策定されミドルウェア実装製品が登場しており、AWS でも適用可能である。しかし、AWS はシステム間で運用が統一されていることを前提としておらず、長期間にわたるセッションの維持や入出力キューの耐久性など既存技術では問題があることが分かっている。そのため、これらの問題を解決したメッセージング基盤を開発した。

2. AWS メッセージング基盤の研究開発

AWS は自律的なシステム内でのビジネスメッセージ支援を目的としており、そのミドルウェアの基盤として非同期メッセージング機能が必要である。AWS では長期間にわたるセッションの維持(VLSession と呼んでいる)、耐久性の高い入出力キューの実装が必要であることが分かっている[1]。そこで、2008 年度よりこれらの問題を解決したメッセージング基盤の本格的実装を開始した[2][3]。しかし、通信の暗号化や認証といったセキュリティ対策、標準プロトコル(ebXML[5])の適用、VLSession の一時停止/再開をするアーカイブ機能などの課題が残っていた[4]。そこで第 3 次プロトタイプでは、サーバ/サーバ間でのやり取りを行う Push 型メッセージングとクライアント/サーバ間でのやり取りを行う Pull 型メッセージングの両方に対応したメッセージング基盤にし、今までの課題を解決した AWS メッセージング基盤の開発を行った。

3. AWS メッセージング基盤の全体像

AWS メッセージング基盤の全体像を図 1図 2に示す。AWS メッセージング基盤は、サーバ/サーバ間でやり取りを行う Push 型メッセージングとクライアント/サーバ間でやり取りを行う Pull 型メッセージングをサポートしている。そのため、SOHO(Small Office/Home Office)や SME(Small and Medium Enterprise)、個人、モバイルといったサーバを持たない環境でもメッセージの送受信が行える。

Autonomous Web Services Development of Messaging base

†Taisuke Kimura, Ryota Ninomiya, Masamichi Hiramoto, Makoto Oya, Shonan Institute of Technology

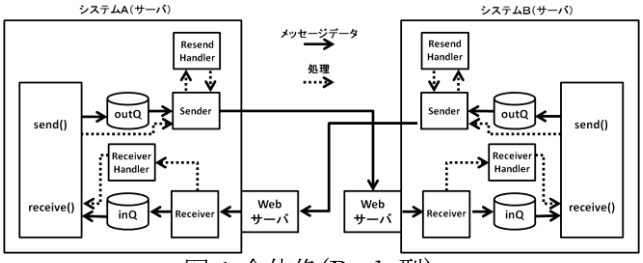

図 1.全体像(Push 型)

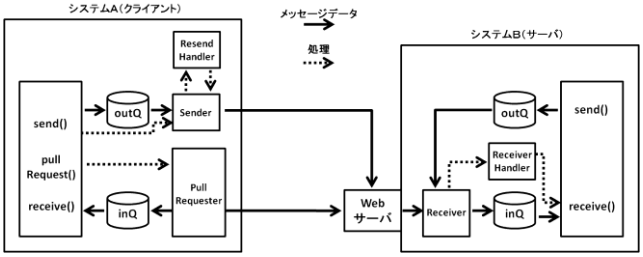

図 2.全体像(Pull 型)

4. AWS メッセージング基盤の構成

4.1. API

API の一覧を表 1 に示す。AWS メッセージング基盤の API は VLSession のクラスメソッドとして実現した。VLSession を開始させるには、startVLSession を使用し自分と相手の URL と UA(ユーザエージェント名;メッセージングのエンドポイントでユニークな名前を持つ)を指定する。メッセージを送信する場合は send を実行する。するとキューに送信メッセージが書き込まれ、送信 ID が返される。receive を実行するとキューから受信したメッセージを取り出す。またタイムアウトを設定できるため、受信できるまで receive でブロック状態にならないようにできる。最後に terminateVLSession を実行し、VLSession を終了させる。send/receive を実行する前には connectDB を実行し、DB に接続しておかなくてはならない。send/receive を長期間実行しない場合、disconnectDB を実行すると DB から接続を解除することができる。DB に接続しているかどうか判断するには isConnectDB を使用し、戻り値が true なら接続中、false なら未接続となっている。

Pull 型メッセージングでのやり取りの際、クライアント側は自分宛のメッセージをあらかじめ自システムのキューに書き込んでおく必要がある。そこで receive 実行前に pullRequest を実行し、自分宛メッセージをあらかじめ受信し、キューへ

書き込んでおく。また、戻り値として受信したメッセージの数が返されるようになっているため、何回 receive を実行する必要があるのか分かるようになっている。

表 1.API 一覧表

名称	説明
startVLsession	VLSession を開始させる
send	メッセージを送信する
receive	メッセージを受信する
pullRequest	メッセージをダウンロードする
terminateVLSession	VLSession を終了する
connectDB	DB に接続する
disconnectDB	DB から接続を解除する
isConnectDB	DB の接続状態を返す

4.2. プロセス/スレッド構造

プロセス/スレッド構造を図 3 に示す。実線は各プロセスとの関係を示す。メッセージの送信(Sender)、再送管理(ResendHandler)、受信(Receiver)、受信管理(ReceiverHandler)およびメッセージのダウンロード機能(PullRequester)の 5 つのプロセスに通信制御機能を分け、各々の制御の効率化、非同期処理できる構造にした。Sender はメッセージを HTTP/SOAP で送信、ResendHandler は再送となったメッセージの管理、PullRequester は自分宛メッセージのダウンロード、Receiver はメッセージの受信および PullRequester からのダウンロード要求の処理、ReceiverHandler は受信したメッセージの管理を行う。各プロセスは DB へのメッセージ格納完了または再送情報を連絡する必要がある。このプロセス間連絡は UDP を用いて実現した。

Sender と ReceiverHandler、PullRequester は複数の VLSession および Receiver からの要求処理をしなければならない。そこでマルチスレッドにし、要求ごとに処理できるようにした。また、リソースを有効に使用できるようスレッドプールを用いている。

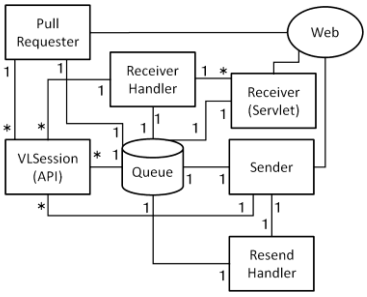

図 3.プロセス/スレッド構造

4.3. 入出力キュー

DB を用い、信頼性・耐久性の高い入出力キューを実現した。

5. セキュリティ対策

AWS メッセージング基盤は以下のセキュリティ対策が必要である。

- メッセージの暗号化
- 通信相手の認証

AWS メッセージング基盤は HTTP を用いた通信のため、SSL を使用しメッセージを暗号化する。通信相手の認証は、クライアント証明書を用い信頼している相手と通信をできるようにする。

6. 性能評価

測定結果のグラフを図 4 に示す。測定方法は、Push 型メッセージングで 1KB、5KB、10~90KB、100~900KB、1~10MB のテキストファイル 30 個をそれぞれ 10Mbps、100Mbps、1Gbps の通信速度で 100 回ずつ送受信を行い、最大/最小時間を除き平均をとった。

AWS では、主に 1KB から 1MB の間のファイルデータを送受信されると予想される。10Mbps では約 260～4000 ミリ秒、100Mbps および 1Gbps では約 260～1700 ミリ秒、送受信にかかっている。この結果から、大きな遅延も見られず実用可能範囲であると言える。

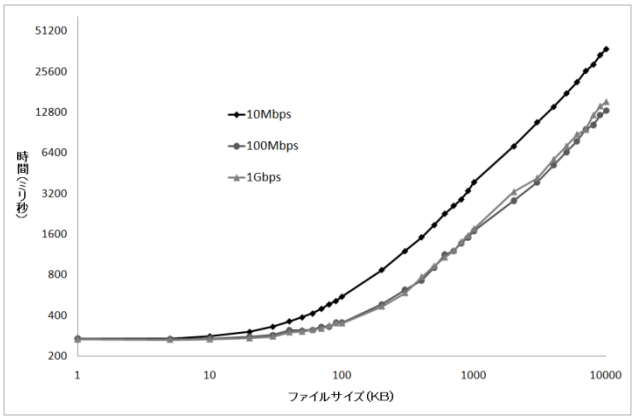

図 4.測定結果

7. まとめと課題

本論文で述べたとおり Push 型と Pull 型両方に対応した第 3 次プロトタイプの開発を行った。また動作テストをし、正しく動作することを確認した。しかしセキュリティ対策、ebXML の準拠、送信エラーとなったメッセージの処理、メッセージング基盤全体のアーカイブなどの課題が残っており、今後これらの課題を解決し AWS メッセージング基盤を完成させる必要がある。

本研究は科研費(21500110)の助成を受けたものである。

参考文献

1. 大谷、伊東他、AWS(自律型 Web サービス)とそのミドルウェア、情報処理学会第 71 回全国大会講演論文集、pp.1-503-504、2009
2. 木村、高木他、AWS ミドルウェアの研究-自律型メッセージング層-、情報処理学会第 71 回全国大会講演論文集、pp.1-515-516、2009
3. 吉川、木村他、AWS における非対称構成型メッセージング機能の実現、情報処理学会 72 回全国大会論文集、pp.1-795-796、2010
4. 木村、吉川他、AWS メッセージング基盤改良の検討、情報処理学会第 72 回全国大会論文集、pp.1-793-794、2010
5. OASIS, ebXML Messaging Services Ver.3.0: Part 1. Core Features, OASIS Standard, 2007